Claims

1. A method of editing an initial image, comprising the steps of;
   - A first process requesting a filter from a second process;
   - Said first process defining a relationship between said filter and said initial image, said related filter and initial image comprising a program,
   - Said second process compiling said program, yielding a compiled program;
   - Running at least a portion of said compiled program to apply a function of said filter to said image, yielding an pixel-image result
2. The method of claim 1 having the additional step of
   - Optimizing said program.
3. The method of claim 2, wherein the step of optimizing includes the step of using a cache look-up to see if said pixel-image result is already in cache.
4. The method of claim 2, wherein the step of optimizing includes the step of calculating an intersection, said intersection representing an area where said pixel-image result is both defined and in a result region requested of said second process.
5. The method of claim 4 further comprising the step of, using said calculated intersection to limit the number of pixels that require calculation during said running of said compiled program.
6. The method of claim 4 further comprising the step of, using said calculated intersection to limit the amount of memory necessary for storing calculated images.
7. The method of claim 1 wherein said compiled program is for a single microprocessor.
8. The method of claim 1 wherein said compiled program comprises a component for a first microprocessor and a component for a second microprocessor.
9. The method of claim 7 wherein said single microprocessor is a CPU.
10. The method of claim 7 wherein said single microprocessor is a programmable GPU.
11. The method of claim 8 wherein said first microprocessor is a CPU and said second microprocessor is a GPU.
12. The method of claim 8 wherein said first and second microprocessors are both CPUs.
13. The method of claim 8 wherein said first and second microprocessors are both GPUs.
14. The method of claim 1 wherein said initial image is only a color.

15. The method of claim 1 wherein said first process is an application program.

16. The method of claim 1 wherein said second process comprises a suite of graphics services functions.

17. The method of claim 1 wherein an operating system comprises said second process.

18. The method of claim 1 wherein said first process requests a high-level filter and said second process responds with an object representing a lower-level filter.

19. The method of claim 1 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.

20. The method of claim 2 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.

21. The method of claim 5 wherein said first process and second process run on a CPU and said compiled program runs on a GPU.

22. A system for editing an initial image, comprising:
- A first microprocessor running a first process and a second process for servicing requests from said first process;
- A memory for storing a filter, said filter requested by said first process;
- A second memory for storing a data structure comprising a relationship between said initial image and said filter, said first process causing the creation of said data structure;
- A second microprocessor for running a program created using said data structure;
- A third memory for storing a pixel image resulting from running said program.

23. The system of claim 22 wherein said first and second memories are the same.

24. The system of claim 22 where in said first, second and third memories are the same.

25. The system of claim 22 wherein said first and second memories are in system memory and said third memory is in video memory.

26. The system of claim 22 wherein said first microprocessor processes said data structure to produce said program for use on said second microprocessor.

27. The system of claim 22 wherein said second microprocessor, under control of said program, causes said pixel image to be stored in said third memory.

28. A method of creating an result image comprising the steps of:

- a first process requesting the creation of a context;
- said first process requesting the creation of a result image;
- said first process indicating parameters associated with the creation of said result image;
- said first process requesting that said result image be rendered to said context;
- a second process servicing said requests of said first process, said servicing comprising the steps of:
    - optimizing a graph representing said result image;
    - compiling said optimized graph;
    - causing the rendering of said compiled optimized graph into said context.

29. The method of claim 28 wherein said creation of said result image comprises editing a pre-existing image.

30. The method of claim 28 wherein said step of optimizing a graph representing said first image comprises the step of:
- Calculating an intersection, said intersection representing an area where said result image result is both defined and requested as a result by said first process.

31. The method of claim 28 wherein said step of optimizing a graph representing said first image comprises the step of:
- Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

32. The method of claim 30 wherein said step of optimizing a graph representing said first image comprises the additional step of:
- Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

33. The method of claim 31 wherein the step of analyzing adjacent nodes comprises the step of checking a cache to determine if the result of such analysis is available in a memory.

34. The method of claim 28 wherein the step of optimizing said graph comprises the step of checking a cache to determine if said graph has already been optimized.

35. The method of claim 28 wherein the step of optimizing said graph comprises the step of checking a cache to determine if the result of rendering said graph is available in a memory.

36. The method of claim 28 wherein said first process requests the output of a graph programmatically assembled by said first process and comprising one or more pre-defined filters.

37. The method of claim 36 wherein said first process programmatically assembled said graph in cooperation with said second process.

38. The method of claim 28 wherein said first process is an application program.

39. The method of claim 28 wherein said second process comprises a suite of graphics services functions.

40. The method of claim 39 wherein an operating system comprises said second process.

41. The method of claim 28 wherein an operating system comprises said second process.

42. The method of claim 28 wherein said second process inserts nodes into said graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme.

43. A method of creating an result image comprising the steps of:

- a first process running on a first microprocessor requesting the creation of a context;
- said first process requesting the creation of a result image;
- said first process indicating parameters associated with the creation of said result image;
- said first process requesting that said result image be rendered to said context;
- a second process running on said first microprocessor servicing said requests of said first process, said servicing comprising the steps of:
    - optimizing a graph representing said result image;
    - compiling said optimized graph;
    - causing rendering of said compiled optimized graph into said context, said rendering occurring on a second microprocessor.

44. The method of claim 43 wherein said creation of said result image comprises editing a pre-existing image.

45. The method of claim 43 wherein said step of optimizing a graph representing said first image comprises the step of:

- Calculating an intersection, said intersection representing an area where said result image result is both defined and requested by said first process.

46. The method of claim 43 wherein said step of optimizing a graph representing said first image comprises the step of:

- Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

47. The method of claim 45 wherein said step of optimizing a graph representing said first image comprises the additional step of:

- Analyzing adjacent nodes in said graph for the purpose of attempting to consolidate nodes.

48. The method of claim 46 wherein the step of analyzing adjacent nodes comprises the step of

- checking a cache to determine if the result of such analysis is available in a memory.

49. The method of claim 43 wherein the step of optimizing said graph comprises the step of

- checking a cache to determine if said graph has already been optimized.

50. The method of claim 43 wherein the step of optimizing said graph comprises the step of

- checking a cache to determine if the result of rendering said graph is available in a memory.

51. The method of claim 43 wherein said first process requests the output of a graph programmatically assembled by said first process and comprising one or more pre-defined filters.

52. The method of claim 51 wherein said first process programmatically assembled said graph in cooperation with said second process.

53. The method of claim 43 wherein said first process is an application program.

54. The method of claim 43 wherein said second process comprises a suite of graphics services functions.

55. The method of claim 54 wherein an operating system comprises said second process.

56. The method of claim 43 wherein an operating system comprises said second process.

57. The method of claim 43 wherein said second process inserts nodes into said graph for converting an original color scheme to an operating color scheme and for converting the operating color scheme back to the original color scheme.

58. A method for providing a high level interface to a graphics processing resource comprising:

- A first process requesting performance of a task through one or more function calls serviced by said graphics processing resource, said function calls selected from one or more of the following options, (i) creating of an image, (ii) creating a filter, (iii) setting arguments associated with said filter, (iv) asking for the output of said filter or another filter, (v) creating a context; and (v) rendering a image to said context or another context;
- Said request having an object associated therewith, said object comprising one of the following: a context, an image, a filter, or a vector;
- Said request defining a relationship between at least one of said functions and one of said objects,
- Said graphics processing resource servicing said request.

59. The method of claim 58 wherein said request is serviced using a GPU and a CPU.

60. The method of claim 58 wherein said request is serviced using an emulator to run a GPU program on a CPU.

61. The method of claim 58 comprising the additional step of creating a graph representing an image.

62. The method of claim 61 comprising the additional step of optimizing said graph.

63. A method for providing a high level interface to a graphics processing resource comprising:
- A first process requesting performance of a task through one or more function calls serviced by said graphics processing resource, said function calls selected from one or more of the following options, (i) creating of an image, (ii) creating a context; and (v) rendering a image to said context or another context;
- Said request having an object associated therewith, said object comprising one of the following: a context, or an image
- Said request defining a relationship between at least one of said functions and one of said objects,
- Said graphics processing resource servicing said request.

64. The method of claim 63 wherein said request is serviced using a GPU and a CPU.

65. The method of claim 63 wherein said request is serviced using an emulator to run a GPU program on a CPU.

66. The method of claim 63 comprising the additional step of creating a graph representing an image.

67. The method of claim 66 further comprising the step of optimizing said graph.

68. The method of claim 63, wherein said function calls include the option of creating a filter.

69. The method of claim 63 wherein said function calls include the option of setting arguments associated with said filter.

70. The method of claim 68 wherein said function calls include the option of setting arguments associated with said filter

71. The method of claim 63 wherein said another object associated with said request may comprise a filter

72. The method of claim 63 wherein another object associated with said request may be a vector.

73. The method of claim 71 wherein another object associated with said request may be a vector.

74. A method for rendering an image comprising the steps of,

- A first process running on a CPU requesting the creation of an image;
- A graphics services resource, responding to said request by running a first routine on said CPU, said first routine for optimizing a graph representing said image;
- Said first process requesting the rendering of said image to a specified context;
- Said graphics services resource causing the rendering of said graph representing said image, said rendering occurring on a GPU.

75. A method for converting a first image representation into a second image representation, comprising the steps of

- Creating a first graph associated with said first image representation where software routines for creating such graph execute on a CPU,
- Determining an intersection of the first graph's global domain of definition and global region of interest;
- Resolving a first node in said first graph by running software routines on said CPU to (i) determine if said first node may be combined with a second node and (ii) create program steps for calculating and storing only the portions of any intermediary mage

that relate to said intersection, said program steps for compilation on said CPU and execution on a GPU.

76. The method of claim 75 wherein the step of determining if said first node may be combined with said second node comprises the step of checking a cache to determine if there is a result in memory regarding such determination regarding combining nodes.

77. The method of claim 75 wherein the step of resolving said first node comprises the step of checking a cache to determine if there is a result in memory regarding the resolution of said first node.

78. The method of claim 75 wherein said first node is a root node.

79. An image processing application program interface embodied on one or more computer readable media, comprising: a first group of services related to filter objects; a second group of services related to image objects; a third group of services related to context objects; and a fourth group of services related to vector objects.

80. The image processing application program interface of claim 79, wherein the first group of services comprise: first functions to create filter objects; second functions to set filter object parameters; and third functions to obtain filter object output.

81. The image processing application program interface of claim 79 wherein the second group of services comprise: first functions to create image objects; and second functions to render an image object to a context object.

82. The image processing application program interface of claim 79, wherein the third group of services comprise first functions to create context objects.

83. The image processing application program interface of claim 79 wherein the fourth group of services comprise: first functions to create vector objects.

84. An application program interface for facilitating image processing, the application program interface comprising functions to: create image objects; create context objects; create filter objects; set filter object parameters; obtain filter object output; and convert image objects into context objects.

85. A computer-readable medium having computer executable instructions for performing the method recited in any one of claims 1, 28, 43, 58, 63, 74 or 75.